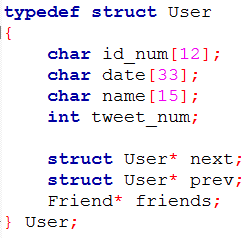
자료구조 final\_assignment - manual

2015410048 장용현

구조체는 총 3가지가 있다.

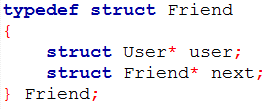
Friend, Word, User인데 각각 친구관계, tweet, 유저 프로필을 저장하고 연결리스트로 관리된다.

* User



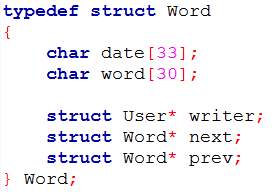
next와 prev를 이용해서 삭제와 추가가 편하게 만들었고 friends는 해당 유저와 친구인 유저들을 저장한다.

* Friend



User는 친구의 유저 프로필을 가리키고 이를 이용해서 친구 리스트에서 친구로 바로 접근이 가능하게 했다.

* Word



글에서 글을 쓴 사람으로 접근하기 쉽도록 writer를 만들었고 그 외에는 기본적인 이중 연결 리스트 형식으로 만들었다.

* 파일 입력

void RoadFile\_user(char user[])

void RoadFIle\_friend(char file[])

void RoadFile\_word(char word[])

를 이용해서 리스트에 하나씩 추가하는 식으로 구현하였습니다.

* Total users

RoadFile\_user에서 노드를 추가할 때 마다 total\_user\_num의 값을 증가시켜 얻었습니다.

* Total friendship records

RoadFile\_friend에서 파일의 줄의 수를 세고 3으로 나눠서 구했다.

* Total tweets

RoadFile\_word에서 노드를 추가할 때 마다 체크해서 구했다.

* Average number of friends

=(total\_friendship\_record\*2.0)/total\_user\_num 공식을 이용했다.

* Minimum friends

getMinFriendNum()함수를 이용해서 구했다.

* Maximum number of friends

getMaxFriendNum()함수를 이용해서 구했다.

* Average tweets per user

전체 tweets를 유저 수 만큼으로 나눠서 구했다.

* Minium tweets per user

MinTweet()함수를 이용해서 구했다.

* Maximum tweets per user

MaxTweet()함수를 이용해서 구했다.

Evaluation criteria

2번 메뉴부터 기능을 구현하지 못하였습니다. 하지만 2번의 경우 연결 리스트를 정렬만 할 수 있다면 쉽게 얻어 낼 수 있을 것 같습니다. 시간 복잡도는 삽입 정렬을 이용하면 일반적인 경우에 O(n)을 가진다.

3번 메뉴 역시 삽입 정렬을 이용하면 O(n)의 시간 복잡도로 해결할 수 있다.

4번 메뉴는 검색 알고리즘을 사용하는데 선형 탐색을 이용하면 쉽게 구할 수 있을 것이다.

나머지 5 ~9번의 경우에도 처음 계획에 써둔 것 같은 방법을 이용하면 해결할 수 있을 것이다.

프로그래밍을 하면서 많은 포인터를 사용하였는데 잠깐 한눈 팔면 뭐가 뭐였는지 헷갈리고는 했습니다. 그래서 그림을 그려가며 연결리스트를 표현하고 코드로 구현하는 방식으로 천천히 구현해 나갔습니다. 파일 입력에 관해서는 값이 잘 입력되었는지 확인하기 위해 printf를 이용해 하나씩 확인해보면서 프로그래밍을 했습니다.